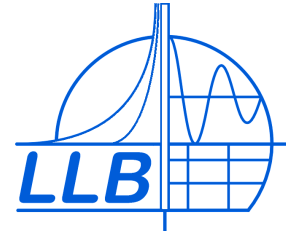


Technische Universität München
Aerospace Department
Institute of Lightweight Structures
Univ. Prof. Dr.-Ing. Horst Baier



LLBo Applets User Guide

Author: Alexei Perro B.Sc.
Supervisor: Erich Wehrle M.Sc.
Date: September 9, 2010

Abstract:

Application of concepts introduced in the course *Multidisciplinary Design Optimization* often requires skills that cannot be acquired from theoretical studies only. This set of interactive applets is meant to illustrate the most fundamental notions in a hands-on way. This guide can serve as a reference while using those applets and also provides some theoretical background on relevant topics.

Keywords:

Minimization, constrained optimization, finite differencing, feasible directions, SUMT, Kuhn-Tucker conditions, fiber reinforced plate, Lagrangian, dual method, ALP

Technische Universität München
Lehrstuhl für Leichtbau
Boltzmannstr. 15
85748 Garching

Contents

List of Figures	1
1 Introduction	3
2 Unconstrained Optimization in One Dimension	5
2.1 Overview	5
2.2 The Graphical User Interface	5
2.3 The Iterative Minimization Procedure	7
2.4 Termination Criteria	8
3 Constrained Optimization in One Dimension	11
3.1 Overview	11
3.2 The Graphical User Interface	11
3.3 The Method of Usable Feasible Direction	13
3.4 Sequential Unconstrained Minimization Technique	13
4 The Kuhn-Tucker conditions in Two Dimensions	15
4.1 Overview	15
4.2 The Graphical User Interface	15
4.3 Optimization of the Fiber Reinforced Plate	17
4.3.1 Load Cases	17
4.3.2 Coordinate System Transformation	18
4.3.3 Constraints	18
4.4 Calculating the Lagrangian Multipliers	19
5 Augmented Lagrangian Procedure	21
5.1 Overview	21
5.2 The Graphical User Interface	21
5.3 The Dual Method	22
5.4 The Augmented Lagrangian Procedure	25
6 Bibliography	29

List of Figures

2.1	Applet window - 1D unconstrained.	6
2.2	One iteration of the iterative minimization procedure.	7
3.1	Applet window - 1D constrained	12
4.1	Applet window - 2D Kuhn-Tucker	16
4.2	Fiber reinforced plate - the two load cases	18
5.1	Applet window - The Augmented Lagrangian Procedure	22
5.2	The Dual Method	24
5.3	The Augmented Lagrangian Procedure	26

1 Introduction

The following sections describe a suite of interactive Java applets designed to support the course *Multidisciplinary Design Optimization*. The sample problems were chosen to make them easy to follow.

Thus, unconstrained optimization is illustrated with an objective function in one variable (Section 2). Then, constraints are added to the same problem (Section 3). For optimization in multiple dimensions, a 2D case is taken as an example (Section 4). In order to better understand the concept of the Lagrangian, a method that uses it directly is also implemented (Section 5).

The illustrative examples enumerated above allow making many important connections to the material of the lectures [1]. Hopefully, experimenting with the applets can be helpful in grasping the practical implications of the theory involved.

Concerning the theory in both its mathematical and its algorithmical aspects, the necessary remarks can be found in the text that follows. Special attention is devoted to those crucial details that are covered only at an abstract level in the lecture script [1].

Of course, this User's Guide cannot provide the overall picture of all the relevant topics. For that, the lecture notes must be consulted. Both detailed and comprehensive treatment of optimization theory aspects can be found in books [3, 4, 5]. References to those are made accordingly.

2 Unconstrained Optimization in One Dimension

2.1 Overview

As an introductory example, the problem of unconstrained minimization in 1D is considered. In the following, the relevant theoretical aspects are outlined and the workings of this particular interactive implementation are explained. The very next section describes available GUI controls and how they help to explore the effects of various decisions in the construction of the minimization procedure. Further, the algorithm of root finding with the method of iterations is explained. In this applet, the minimum is identified as the root of the first derivative (for the details, see [1, §6.4]). Finally, the three available termination criteria are explicated with formulas.

2.2 The Graphical User Interface

The overall view of the applet window is shown in Figure 2.1.

The left part shows a plot of the objective function and on the right there are a number of panes for controlling the optimization procedure.

Procedure control contains two buttons:

- *Minimize* runs the procedure with the parameters currently set and displays the intermediate steps in the plot;
- *Start over* clears the plot to the initial state.

Initial point provides one button and displays a message string. Upon pressing *Set start* you will be offered to specify the initial approximation with a mouse click in the plot.

Objective function offers a choice among the two available objective function.

Gradient calculation allows adjustment of the way in which the derivative of the objective function is evaluated:

- *Analytical*: the formula that results from differentiating the objective function by hand is used.
- *Finite differences*:

$$\text{forward: } y'(x) = \frac{y(x + \Delta x) - y(x)}{\Delta x} \quad (2.1)$$

$$\text{central: } y'(x) = \frac{y(x + \frac{\Delta x}{2}) - y(x - \frac{\Delta x}{2})}{\Delta x} \quad (2.2)$$

$$\text{backward: } y'(x) = \frac{y(x) - y(x - \Delta x)}{\Delta x} \quad (2.3)$$

Step size accepts a value for the parameter α (explained in section 2.3).

Termination criteria offers a selection of termination criteria (explained in Section 2.4).

Optimization ends as soon as *at least one of* the criteria that are *currently checked on* is fulfilled.

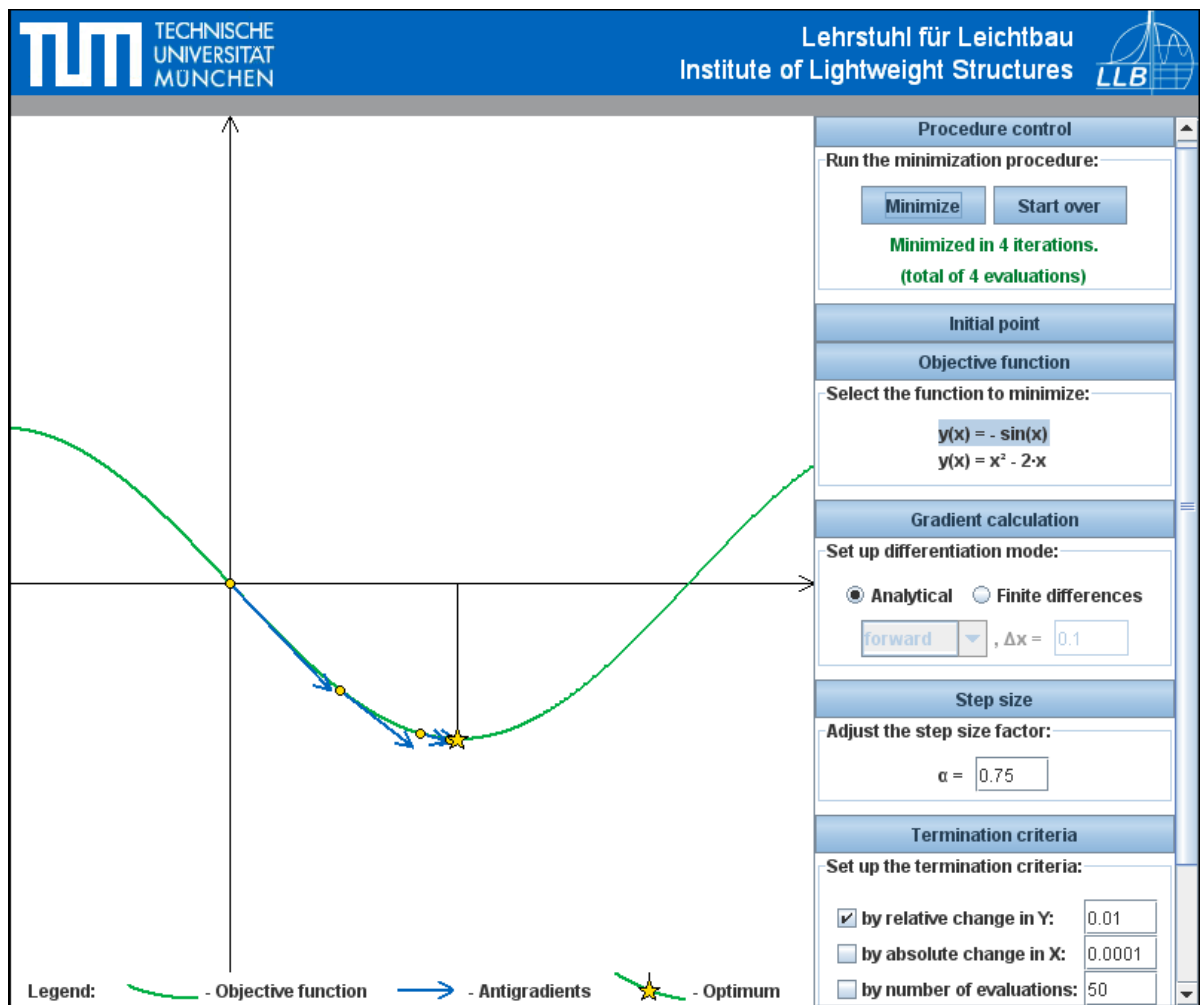


Figure 2.1: Applet window - 1D unconstrained.

2.3 The Iterative Minimization Procedure

The algorithm described below can be viewed as a (simplified) special case of “the steepest descent” in 1D, see [1, §6.4]. For numerical optimization, an initial point x_0 must be selected as a rough approximation. A sequence of increasingly better approximations is generated via the following update rule:

$$x_{i+1} = x_i - \alpha \cdot \nabla y(x_i)$$

The value for x_i when $i = 0$ and the value of α are parameters to the procedure. The derivative ∇y can be evaluated either numerically or analytically, as was mentioned in section 2.2, *Gradient calculation*. The update rule is illustrated in Figure 1.2.

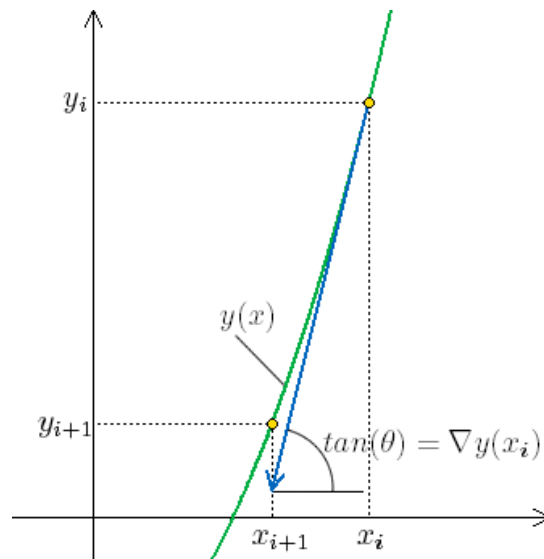


Figure 2.2: One iteration of the iterative minimization procedure.

The overall procedure flow may be summarized in pseudo-code as follows (assuming that the functions y , y_prime , and $converged$ have been suitably defined):

Algorithm 2.1

```

10: minimize_iterative(x0, alpha)
20: i = 0;
30: xi_1 = x0;
40: yi_1 = y(xi_1);
50: do
60:   i = i + 1;
70:   xi = xi_1;
80:   yi = y(xi_1);
100:  xi_1 = xi - alpha * y_prime(xi);
110:  yi_1 = y(xi_1);
120: while not converged(xi, yi, xi_1, yi_1, i);
130: return xi_1;

```

As can be seen, the call to `converged` in line 120 accepts both the old and the updated values of x and y . The next section describes how these values are used to halt the procedure in due time.

2.4 Termination Criteria

Three different criteria are implemented and can be applied either individually or in combination. In every case, the purpose is to halt the optimization procedure when no further progress is envisioned, i.e. when x_i would stay almost constant in further iterations. The criteria are as follows.

Relative change in y :

$$\left| \frac{y_{i+1} - y_i}{y_i} \right| \leq \varepsilon$$

Absolute change in x :

$$|x_{i+1} - x_i| \leq \delta$$

Predefined limit number of evaluations:

$$i \geq i_{max}$$

The parameters ε , δ , and i_{max} can be set independently for every criterion in the user interface. When more than one criteria are checked, a change in a parameter sometimes has no effect on the course of execution. This indicates that the respective criterion does not cause termination of the algorithm in this particular setting, but rather one of the other activated criteria does.

Also note that the effect of i_{max} on the number of iterations changes depending whether the *Finite differences* or the *Analytical* mode is selected in the *Gradient calculation* tab. In the former case, the counter i in line 60 of Algorithm 1.1 is incremented by two units instead of one. The difference in the number of evaluations per iteration should be apparent from formulas 2.1-2.3. Of course, with finite differences function evaluations are counted, not evaluations of the analytical derivative.

3 Constrained Optimization in One Dimension

3.1 Overview

The following sections describe the applet that illustrates two approaches to constrained optimization. The sample problem is identical to the one considered previously (see Section 2), just the constraints are added. Therefore, the user interface contains some additional controls, which are described in the next section.

Sections 3.3 and 3.4 provide references concerning two methods based on the iterative procedure (Section 2.3).

- The method of usable feasible direction is quite a straightforward extension, just one more termination criterion is added. An outline of this principle can be found in [1, §6.6].
- Sequential unconstrained minimization technique (SUMT) constructs an *auxiliary function* that automatically takes the constraints into account (for details please consult [1, §6.2]).

3.2 The Graphical User Interface

The applet window is shown in Figure 3.1 with the additional control pane expanded. The rest of the panes are the same as in LLBoApplet1, please refer to Section 2.2 for details.

The **Methods** pane contains two radio buttons and also the means to fine-tune the factor r :

- *Usable feasible direction* activates the simple way of constraint handling, it has no parameters (see Section 3.3).
- *SUMT* activates Sequential Unconstrained Minimization Technique, which depends on parameter r (see Section 3.4).
- The text field labeled [$r =$] accepts the value for r . The new value is applied upon the next minimization run (invoked via the *Minimize* button in the *Procedure control* pane).

- The buttons labeled $[/ 10]$ and $[* 10]$ serve to automatically scale r by an order of magnitude with every click. Besides, they simultaneously produce a convenient side effect: the currently achieved minimum is set as the initial approximation x_0 for the next run (see Section 2.3).

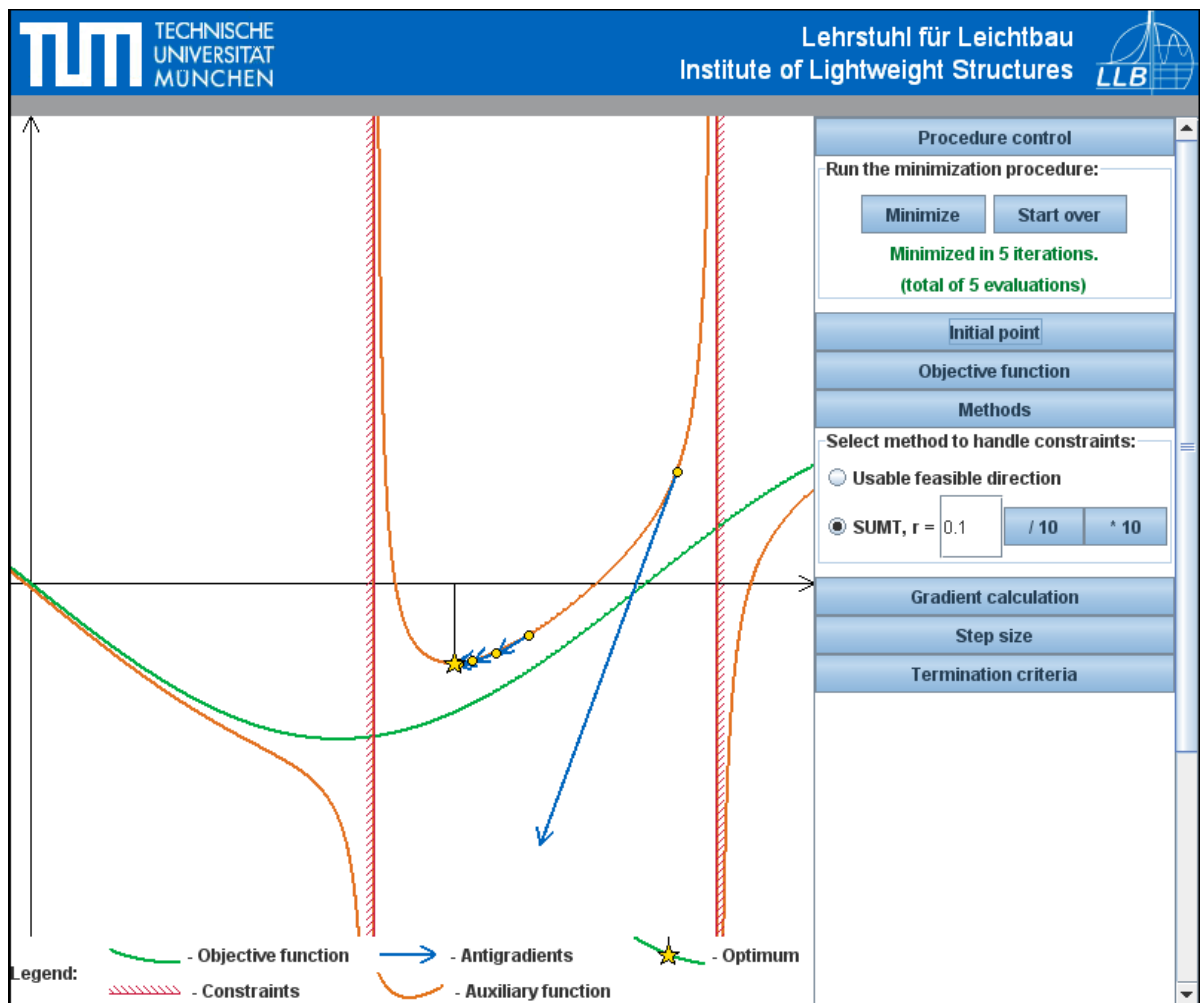


Figure 3.1: Applet window - 1D constrained

3.3 The Method of Usable Feasible Direction

For the 1D setting the method under consideration is trivial. Another termination criterion that is added internally to the algorithm is all that changes compared to the basic steepest descent (see Section 2.3). This additional criterion is simply, “as soon as x_{i+1} lies outside the feasibility region, return x_i as an optimum.” One example of how the same principle is applied in a more sophisticated context is Sequential Quadratic Programming (SQP), see [1, §6.6].

3.4 Sequential Unconstrained Minimization Technique

A more advanced way to handle constraints is to eliminate them altogether through a suitable transformation of the problem: from the objective function with constraints to an unconstrained *auxiliary function*¹. Here “suitability” means that the solutions of the original and the transformed problems must coincide. The lecture script dwells in quite some detail on how this is achievable with penalty functions, see [1, §6.2] for the mathematical and the algorithmical details. Just two more points are to be made in connection with this specific implementation:

1. For illustrative purposes, the algorithm is left to be “semi-automatic”, i.e. steps 4 and 5 in [1, p. 76] are left to be triggered by the user with the button labeled [/ 10] (found in the *Methods* pane).
2. As mentioned in the script, “the conditioning of the replacement function becomes increasingly worse as r becomes smaller” ([1, p. 76]). Indeed, this may manifest itself as an error message reading ‘*The procedure led into an infeasible region*’ in the *Procedure control* pane. This would indicate that the initial approximation is unsuitable, given the current value for α . In order to keep to the feasibility region, α must be decreased using the Step size pane. However, with decreased α the number of steps until convergence may become unreasonable. This in turn can be helped through somewhat relaxing the termination criteria (see Section 2.4).

As can be noticed, in practice this more sophisticated constraint handling strategy may involve certain subtleties. Experimenting with the sample problem can help revealing them. For thorough understanding based on theory, a mathematically rigorous treatment of SUMT may be found in [3].

¹Also called “*replacement function*” in [1].

4 The Kuhn-Tucker conditions in Two Dimensions

4.1 Overview

The applet described below illustrates some fundamental aspects of optimization in multiple dimensions. To keep the example comprehensible, a problem in just two dimensions is considered.

The very next section describes the user interface, which displays the 2D space of design variables, the objective function, the constraints, and some supplementary visual clues.

Further, the design problem to be considered is introduced (Section 4.3). Evaluation and analysis of the Kuhn-Tucker conditions (see [1, §3.6]), which form the essence of this applet, is discussed last (Section 4.4).

4.2 The Graphical User Interface

The overall view of the applet window is shown in Figure 4.1. A 2D plot on the left represents the design space in variables α and d (see Section 4.3 for a reference to the problem statement). A click in the plot selects the point to analyse. On the right there are three panes that display characteristics of the currently selected design space point.

Lagrangian multipliers displays the following values once they are calculated:

$\lambda_{1,2}$	The values of the Lagrangian multipliers selected through application of the least squares method, see Section 4.4 for details.
L	The value of the selected Lagrangian at the current point.
$\nabla_{x_{1,2}} L$	The gradient of the selected Lagrangian ¹ at the current point (allows judging whether the point can be an optimum).

¹Note that the multipliers $\lambda_{1,2}$ are calculated to be the best in the least squares sense. The values thus obtained differ over the neighbourhood of a given point in the design space. It should be noted that both the *value* and the *gradient* of the Lagrangian depend on these multipliers. Essentially, a *different function* L is considered in every design point. Therefore, the habitual relation between function derivatives and how the function itself varies cannot be expected to hold in this particular case.

- $g_{1,2}$ The values of the constraint functions (allow judging whether the point lies within the feasibility region).
- $x_{1,2}$ The values of the design variables (x_1 and x_2 correspond to α and d , respectively).

The *Calculate multipliers* button makes the current point fixed and triggers calculation of the values enumerated above. The *Reset* button produces the reverse effect, the values disappear and the selected point again traces the mouse pointer freely, thus updating the *Plate design* pane.

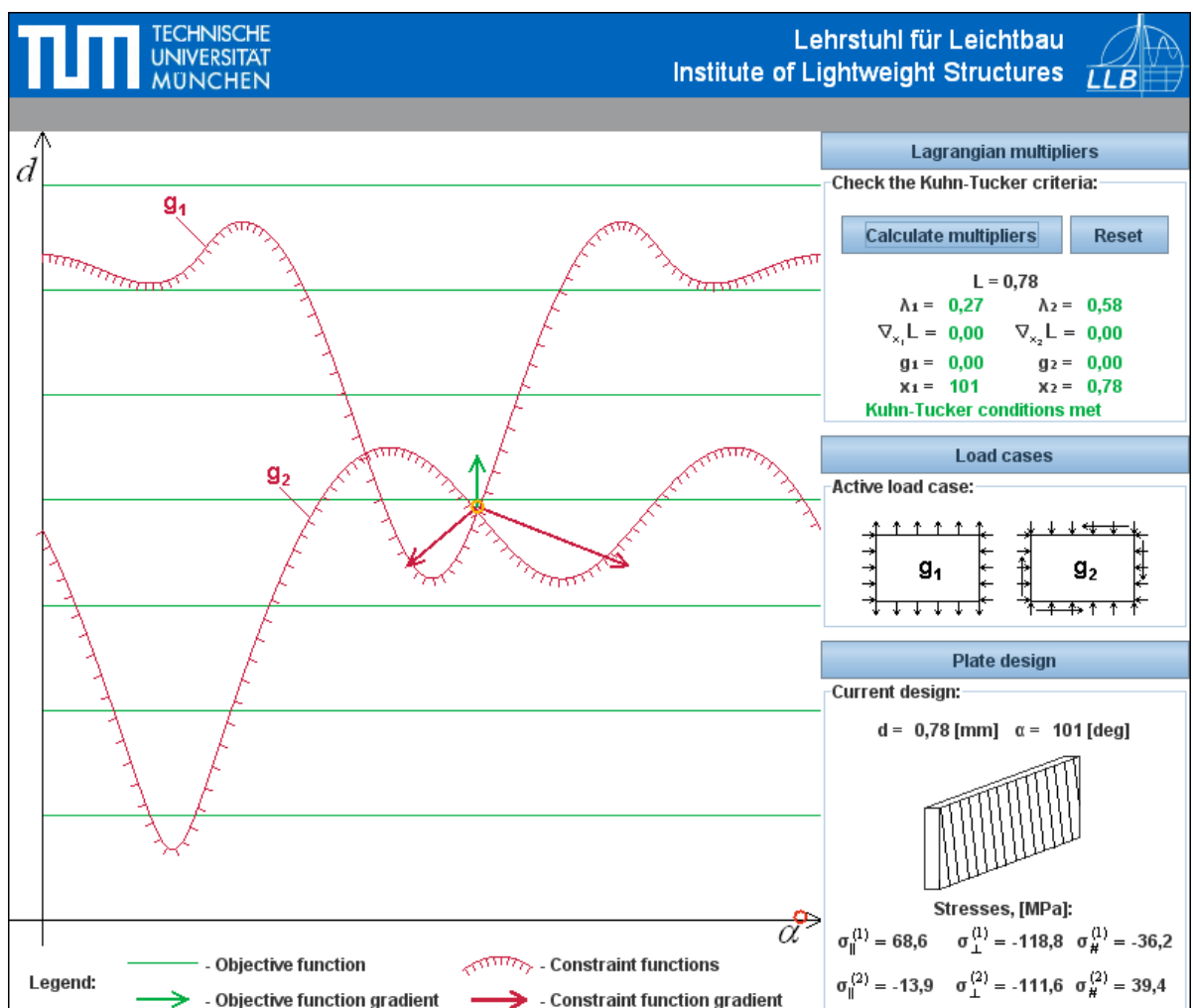


Figure 4.1: Applet window - 2D Kuhn-Tucker

Load cases contains a pair of mnemonic images that represent load cases considered in the problem (see Section 4.3). When the design point is fixed, the active load case(s) are black; otherwise - grayed out.

Plate design shows a 3D sketch of a fragment of the plate, the values of design variables, and also six values for in-plane stresses resulting from the design choice. A triple of stresses is defined for each of the two load cases. The subscript signs have the following meanings:

$\sigma_{ }$	normal stress parallel to the direction of the fibers;
$\sigma_{\perp}$	normal stress perpendicular to the direction of the fibers;
$\sigma_{\#}$	in-plane shear stress.

4.3 Optimization of the Fiber Reinforced Plate

A classical design optimization problem is considered:

$$\min \{w(d) \mid g_1(\alpha, d) \leq 0, g_2(\alpha, d) \leq 0\}$$

Here the weight w to be minimized depends only on the thickness of the plate d . The constraints g_1 and g_2 correspond to load cases described in the next section. These constraints heavily depend on the angle α of the fibers, as becomes clear from sections 4.3.2 and 4.3.3.

Statement and solution of the problem under consideration is taken from work [2]. Before delving into the specific details of this statement expounded below, it is advisable to review a somewhat more accessible formulation. The lecture script [1, §2.1.3] can be helpful in gaining an overall picture.

4.3.1 Load Cases

The specific load values that were used as an example in this implementation are given in Figure 4.2. Regarding load case g_2 it can be observed that due to the choice of values the principal stresses are compressive only. This observation is relevant for what will be discussed further, since in the definition of this problem's constraints, compression and tension are distinguished in a special way (discussed in Section 4.3.3).

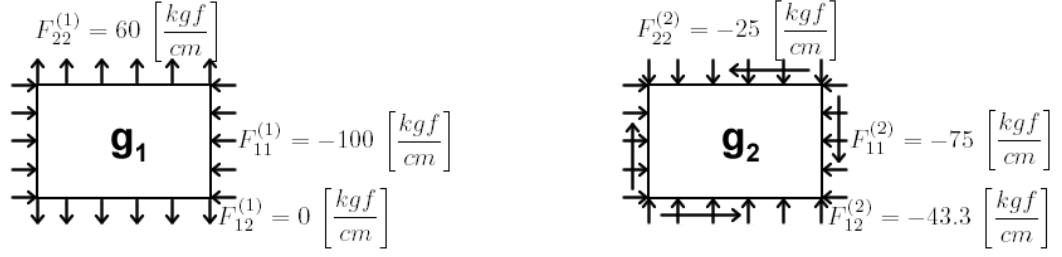


Figure 4.2: Fiber reinforced plate - the two load cases

4.3.2 Coordinate System Transformation

The loads shown in Figure 4.2 are expressed as forces distributed over the length of the edges of a plate element. In other words, integration of stresses over plate thickness has been performed. The constraints that will be discussed in a moment are formulated in terms of stresses $\sigma_{||}$, $\sigma_{\perp}$, and $\sigma_{\#}$. Therefore, the values of distributed forces are to be divided by thickness d in order to obtain stresses. Moreover, the stresses must be transformed to the coordinate system that is aligned to the direction of the fibers (in other words, the system is rotated by angle α).

The needed transformation involves two rotations. First, it is necessary to calculate horizontal and vertical components of stresses at the edges of a *rotated element* of the plate. Second, these components must be expressed relative to the *rotated axes* of the new coordinate system. This explains why the transformation matrix in (4.1) has products of trigonometric functions as its elements.

Uniting all that was said so far, the following formula allows arriving from distributed forces (defined by the load cases) to fiber-aligned stresses (used for defining the constraints):

$$\begin{bmatrix} \sigma_{||} \\ \sigma_{\perp} \\ \sigma_{\#} \end{bmatrix} = \frac{1}{d} \cdot \begin{bmatrix} \cos^2 \alpha & \sin^2 \alpha & 2 \cdot \sin \alpha \cdot \cos \alpha \\ \sin^2 \alpha & \cos^2 \alpha & -2 \cdot \sin \alpha \cdot \cos \alpha \\ -\sin \alpha \cdot \cos \alpha & \sin \alpha \cdot \cos \alpha & \cos^2 \alpha - \sin^2 \alpha \end{bmatrix} \times \begin{bmatrix} F_{11} \\ F_{22} \\ F_{12} \end{bmatrix} \quad (4.1)$$

As can be seen, all derived quantities are obtained analytically. However, the expressions became quite involved already. More complications are added at the stage of defining the constraints, which is discussed next.

4.3.3 Constraints

The requirement that the fibers must not suffer damage is formulated in terms of the boundary allowable stress $\sigma_{||B}$ as follows:

$$\left(\frac{\sigma_{||}(\alpha, d)}{\sigma_{||B}} \right)^2 - 1 \leq 0 \quad (4.2)$$

Moreover, fibers may not become detached from the plate:

$$\left(\frac{\sigma_{||}(\alpha, d)}{\sigma_{||B}} \right)^2 + \left(\frac{\sigma_{\perp}(\alpha, d)}{\sigma_{\perp B}} \right)^2 + \left(\frac{\sigma_{\#}(\alpha, d)}{\sigma_{\#B}} \right)^2 - 1 \leq 0 \quad (4.3)$$

From algebraic considerations it is clear that whenever (4.3) holds, (4.2) holds as well. Therefore, only (4.3) is taken into account.

As for the boundary allowable stresses, it must be noted that their magnitudes are changed whenever the sign of the respective fiber-aligned stress changes. This reflects the fact that materials withstand tension and compression not equally well. In this particular example, the boundary values are as follows (in [MPa]):

$$\sigma_{||B} = \begin{cases} 200 & \text{for } \sigma_{||} > 0 \\ 75 & \text{for } \sigma_{||} < 0 \end{cases} \quad (4.4)$$

$$\sigma_{\#B} = 6.1 \quad (4.5)$$

$$\sigma_{\perp B} = \begin{cases} 4.8 & \text{for } \sigma_{\perp} > 0 \\ 15.5 & \text{for } \sigma_{\perp} < 0 \end{cases} \quad (4.6)$$

To summarize, three sources of nonlinearity are present: coordinate system transformation (4.1), requirement (4.3), and parameters (4.4, 4.6). When acting in sequence these result in the complicated shape of the feasibility region, which can be observed in Figure 4.1.

4.4 Calculating the Lagrangian Multipliers

The Kuhn-Tucker conditions are introduced in [1, §3.6] as “a generalization of the Lagrange multiplier theory that is true for extrema of functions with equality restrictions.” The construction of this generalization is explicated in [4, Chapter 5]. There, additional mathematical apparatus is used, e.g. *slack variables*, the notion of *regular points*, etc. With that, it is possible to formulate the conditions also for the case of inequalities. Besides, they can be made meaningful anywhere in the design space and not just at the bounds of the feasibility region.

Another thing that becomes possible with the extended formulation is obtaining the Lagrangian multipliers regardless of whether the point $\mathbf{x}$ under consideration is an optimum

or not. The vector λ is calculated under an *a priori assumption* that the primary condition holds:

$$\frac{\partial L}{\partial x_i} = \frac{\partial f}{\partial x_i} - \sum_j \lambda_j \frac{\partial g_j}{\partial x_i} = 0$$

Or, in matrix form:

$$\nabla \mathbf{f} - N\lambda = 0 \quad (4.7)$$

Here $\nabla \mathbf{f}$ is the gradient of the objective function at point $\mathbf{x}$, and N is defined by $n_{ij} = \frac{\partial g_j}{\partial x_i}$.

The following formula from [4] approximates λ as a solution to (4.7) in the least squares sense:

$$\lambda = (N^T N)^{-1} N^T \nabla \mathbf{f} \quad (4.8)$$

Once the multipliers are calculated, the gradient of the Lagrangian $\nabla \mathbf{L}$ at point $\mathbf{x}$ can be obtained. One of the following things then happens:

1. $\nabla \mathbf{L}$ is zero. This indicates that the formula (4.8) produced an exact solution to equation (4.7), and $\mathbf{x}$ may be an optimum. The rest of the conditions must be checked.
2. $\nabla \mathbf{L}$ is nonzero. This means that the initial assumption was wrong and $\mathbf{x}$ cannot be an optimum.

5 Augmented Lagrangian Procedure

5.1 Overview

The following sections describe the applet illustrating the dual method of constrained optimization. The sample problem is similar to the one considered previously (see Section 3), but there is only one constraint instead of two. The user interface controls are identical to the first three panes in the unconstrained applet (see Section 2.2), and the plots are described in the next section.

Sections 5.3 and 5.4 outline two constrained optimization strategies based on the concept of the Lagrangian. The corresponding mathematical theory can be found in [5, Ch 17].

- The **dual method** is used in this guide for introductory purposes. The applet itself is implemented in a more sophisticated way.
- The **Augmented Lagrangian Procedure** (ALP) is a further development aiming at better stability and faster convergence. The implemented algorithm is given in full detail as a flow chart in Figure 5.3. However, only the most fundamental aspects are thoroughly commented upon.

For a detailed treatment of the topic, please refer to [5].

5.2 The Graphical User Interface

The applet window is shown in Figure 5.1. The left part shows a plot of the objective function and the constraint. On the right there is a 2D plot of level lines representing the corresponding Lagrangian function (see Section 5.3). At the bottom there are three panes for controlling the optimization procedure:

Procedure control contains two buttons:

- *Minimize* runs the procedure with the parameters currently set and displays the intermediate steps in the plot;
- *Start over* clears the plot to the initial state.

Initial point provides one button and displays a message string. Upon pressing *Set start* you will be offered to specify the initial approximation with a mouse click in either of the two plots.

Objective function offers a choice among the two available objective function.

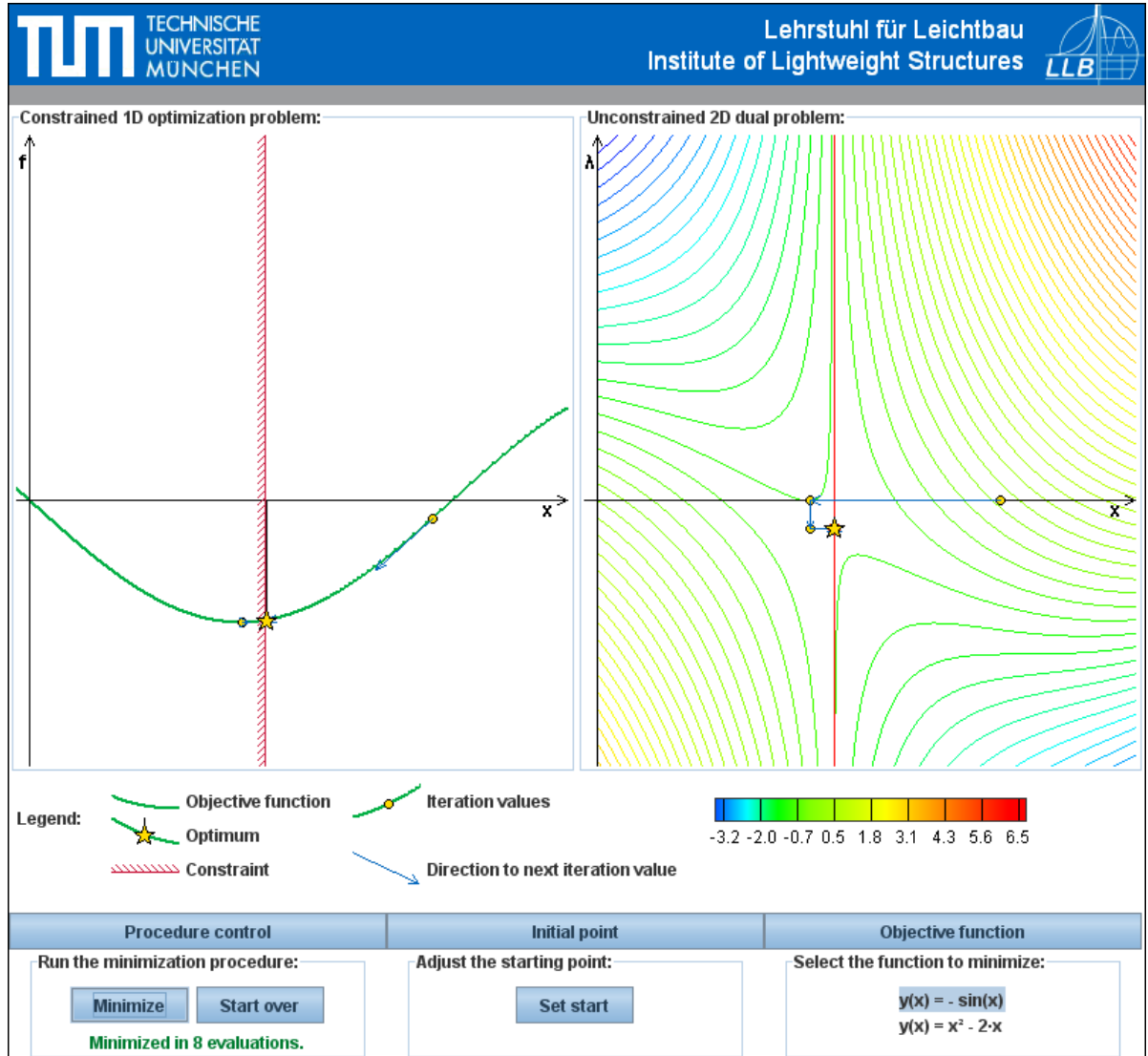


Figure 5.1: Applet window - The Augmented Lagrangian Procedure

5.3 The Dual Method

Along the lines of a more general case [5, §17.1], let us consider the *Lagrangian function* that corresponds to the 1D constrained optimization problem with one *equality* constraint:

$$L(x, \lambda) = f(x) + \lambda g(x),$$

where $f(x)$ is the objective function, $g(x)$ - the constraint, and λ - the Lagrangian multiplier. Now we can define the dual function $D(\lambda)$, which is the minimum of $L(x, \lambda)$ along x when λ is fixed to a certain value as the independent variable:

$$D(\lambda) = \min_x L(x, \lambda)$$

The optimum of the constrained problem is defined as the *saddle point* of the Lagrangian. Using the definitions introduced above, at the optimum it holds:

$$\max_{\lambda} D(\lambda) = \min_x L(x, \lambda)$$

Conventional descent methods are not applicable for finding saddle points. Still, an iterative procedure called the *dual method* can be constructed as a sequence of repeated minimizations while λ is adjusted:

1. Given $L(x, \lambda) = f(x) + \lambda g(x)$, assume some initial $\lambda = \lambda^k$, $k = 0$.
2. Obtain the intermediate optimal solution $x^*(\lambda) = \min_x L(x, \lambda)$.
3. Modify $\lambda = \lambda^{k+1}$.
4. Repeat steps 2 and 3 until convergence.

An open question here is how to modify λ . One possibility is to change it by the value of the constraint g at current point x , multiplied by certain factor $1/\mu$:

$$\lambda^{k+1} = \lambda^k + \frac{g(x^*(\lambda^k))}{\mu}$$

The reasoning behind this formula goes like this, "If constraint g is still violated at the saddle point, this means that the lagrangian multiplier λ is not large enough". In case the iterative process does not converge, this indicates that factor μ should be decreased. The resulting algorithm is presented as a flowchart in Figure 5.2. To make the update rule for λ mathematically sound, we need to augment the Lagrangian function with an extra term, which is discussed in the next section.

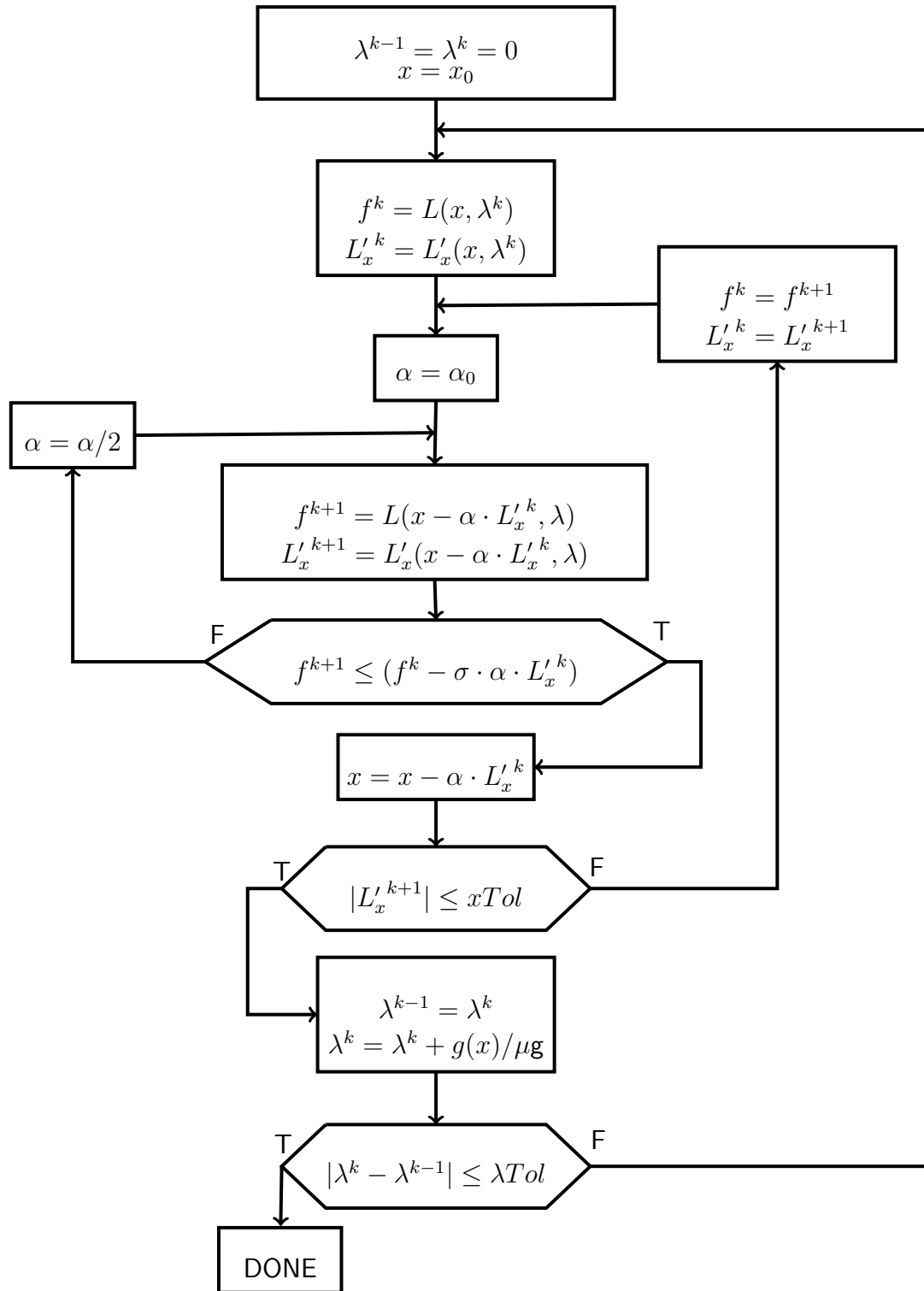


Figure 5.2: The Dual Method

5.4 The Augmented Lagrangian Procedure

The ALP (see [5, §17.5]) is similar to the penalty method in that a penalty term is added (see Section 3.4). But to avoid an ill-conditioned problem in the end, it is added not to the objective function itself but rather to the Lagrangian. If we take g to be again an *equality* for a start, the Lagrangian is augmented as follows:

$$L_A(x, \lambda; \mu) = L(x, \lambda) + \frac{1}{2\mu}g^2(x) = f(x) + \lambda g(x) + \frac{1}{2\mu}g^2(x)$$

From the Kuhn-Tucker condition by certain transformations it follows that near a critical point this holds:

$$\lambda + \frac{g(x)}{\mu} \approx \lambda^*$$

The above suggests an update rule for converging to the correct value for lambda. From this, the following algorithmic framework results:

1. Start with initial approximation x_0, λ_0 , and a chosen factor $\mu_0 > 0$. Request final tolerance tol . For $k = 0, 1, 2 \dots$
2. Starting from x_k , minimize $L_A(x, \lambda_k; \mu)$ until

$$\| \nabla L_A(x, \lambda_k; \mu) \| < \tau_k$$

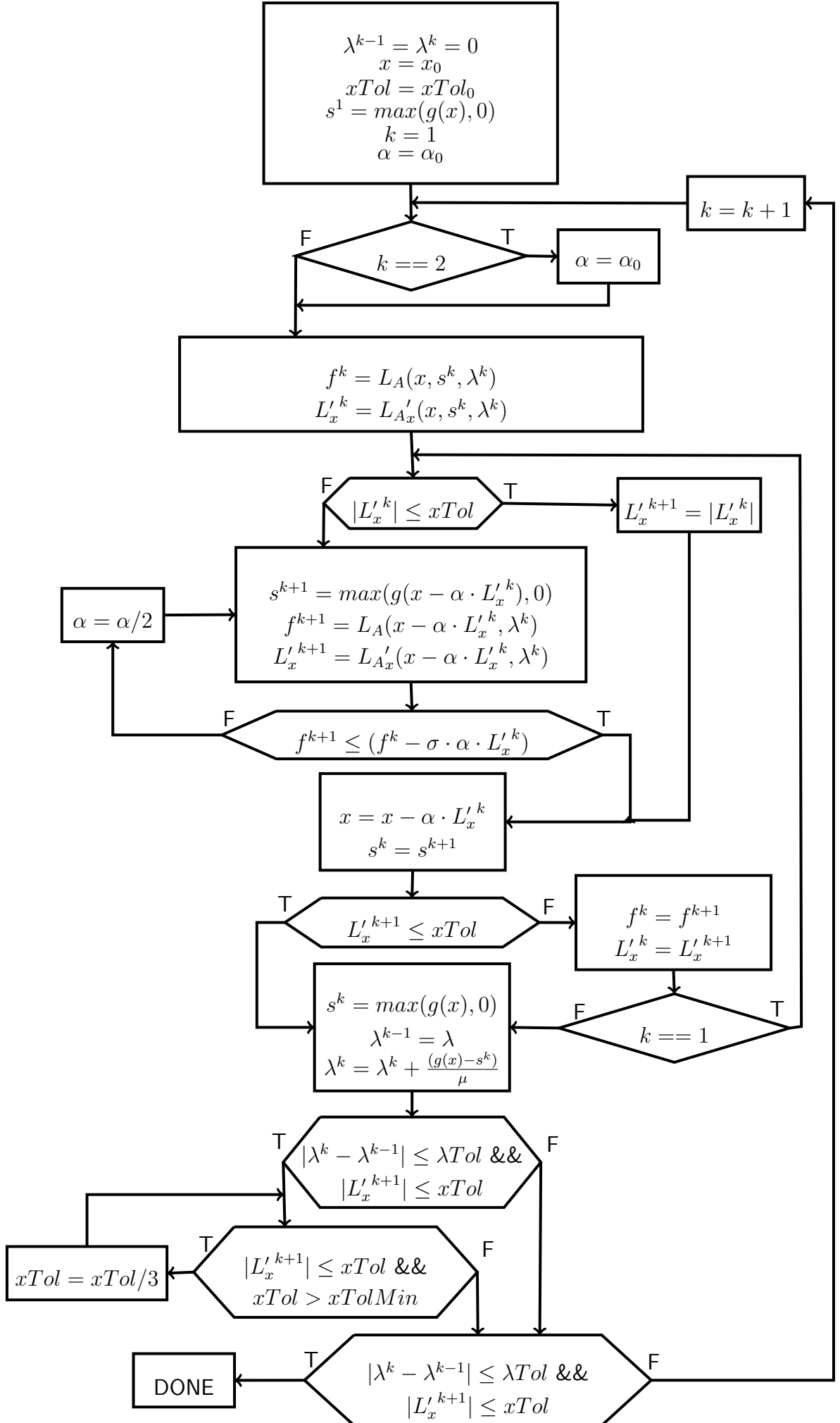
The result of minimization is x_k^* .

3. If $\tau_k < tol$ - we are done!

Else

$$\begin{aligned} \lambda_{k+1} &= \lambda_k + \frac{1}{\mu}g(x_k^*) \\ x_{k+1} &= x_k^* \end{aligned}$$

The flowchart for the resulting algorithm is shown in Figure 5.3.



It can be seen that the flowchart contains a number of algorithmic features in addition to those outlined in the framework. They were added as improvements and are sometimes heuristic in nature. The purpose and main ideas of these enhancements are briefly outlined below:

1. **Slack variable** s is used to account for the fact that g is in fact *inequality* constraint. An expression $(g(x) - s)$ is used in place of $g(x)$, and s is kept equal to $\max(0, g(x))$.
2. **Adjustment of** α is needed to ensure stability of the search along x . Armijo condition (see [5, §3.1]) is used to determine whether α is small enough to produce sufficient decrease in the objective function value, and σ is a parameter to that condition.
3. **Multi-step optimization along x for** $k = 1$ allows to reach good approximation of the intermediate optimum during the first run along x , regardless of the initial x_0 . This allows making just one optimization step along x per every adjustment of λ .
4. **Adjustment of** $xTol$ can result in this parameter changing by as much as three orders of magnitude during an optimization run. It improves efficiency to opt for less precise but faster solutions while the final optimum is far away. As the method converges, the precision is gradually increased.

6 Bibliography

- [1] Baier, Horst; Huber, Martin; Petersson, Ögmundur; and Wehrle, Erich. *Multidisciplinary Design Optimization, Lecture Notes*.
- [2] Baier, Horst; *Mathematische Programmierung zur Optimierung von Tragwerken insbesondere bei mehrfachen Zielen, Dissertation*. Darmstadt, 1978.
- [3] Fiacco, Anthony V. and McCormick, Garth P. *Nonlinear programming: sequential unconstrained minimization techniques*. Philadelphia: SIAM, 1990.
- [4] Haftka, Raphael T. and Gürdal, Zafer. *Elements of structural optimization*. Dordrecht: Kluwer Academic Publishers, 1992.
- [5] Nocedal, Jorge and Wright, Stephen. *Numerical Optimization*. New York: Springer-Verlag, 2000.